



US008473766B2

(12) **United States Patent**
Liu et al.

(10) **Patent No.:** **US 8,473,766 B2**
(45) **Date of Patent:** ***Jun. 25, 2013**

(54) **OPTIMIZING POWER USAGE BY
PROCESSOR CORES BASED ON
ARCHITECTURAL EVENTS**

(75) Inventors: **Yen-Cheng Liu**, Portland, OR (US); **P.
Keong Or**, Portland, OR (US);
Krishnakanth Sistla, Hillsboro, OR
(US); **Ganapati Srinivasa**, Portland, OR
(US)

(73) Assignee: **Intel Corporation**, Santa Clara, CA
(US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **13/609,006**

(22) Filed: **Sep. 10, 2012**

(65) **Prior Publication Data**

US 2013/0031400 A1 Jan. 31, 2013

Related U.S. Application Data

(63) Continuation of application No. 13/329,700, filed on
Dec. 19, 2011, which is a continuation of application
No. 11/618,477, filed on Dec. 29, 2006, now Pat. No.
8,117,478.

(51) **Int. Cl.**
G06F 1/00 (2006.01)

(52) **U.S. Cl.**
USPC **713/323; 713/300; 713/320; 713/322;
713/324; 713/600**

(58) **Field of Classification Search**

USPC 713/300, 320, 322, 323, 324, 600
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,339,445	A *	8/1994	Gasztanyi	713/324
6,908,227	B2 *	6/2005	Rusu et al.	374/141
7,062,304	B2 *	6/2006	Chauvel et al.	455/574
7,146,511	B2 *	12/2006	Barr et al.	713/300
7,174,194	B2 *	2/2007	Chauvel et al.	455/574
7,197,655	B2 *	3/2007	Flachs et al.	713/324
7,222,245	B2 *	5/2007	Singh	713/300
7,272,732	B2 *	9/2007	Farkas et al.	713/320
7,318,222	B2 *	1/2008	Civlin	717/151

(Continued)

FOREIGN PATENT DOCUMENTS

JP	11161383	6/1999
JP	2001229040	8/2001

(Continued)

OTHER PUBLICATIONS

Japanese Patent Office, Office Action mailed Sep. 4, 2012 in Japanese
application No. 2011-138138.

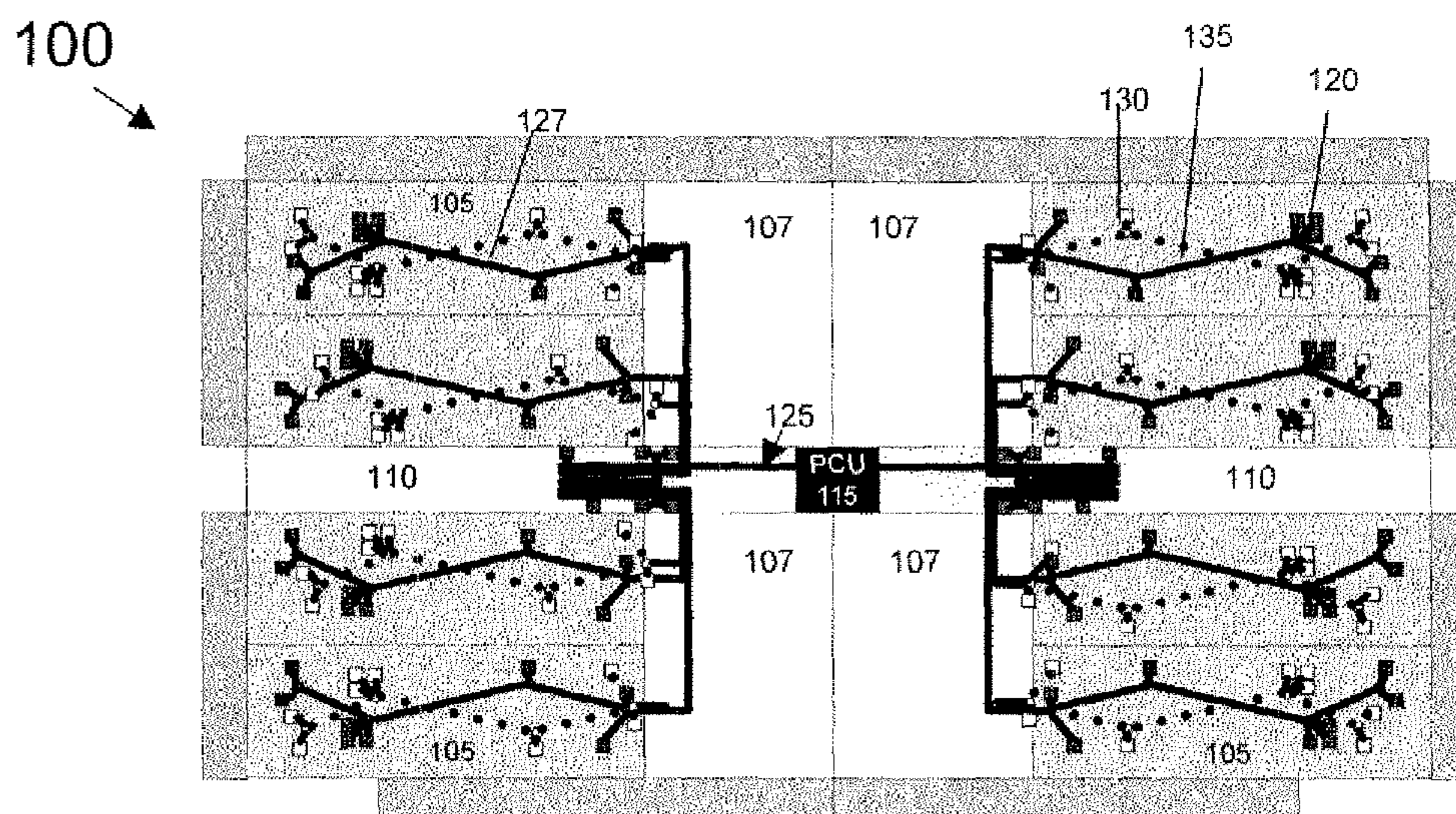
(Continued)

Primary Examiner — Jaweed A Abbaszadeh

(57) **ABSTRACT**

A method and apparatus to monitor architecture events is disclosed. The architecture events are linked together via a push bus mechanism with each architectural event having a designated time slot. There is at least one branch of the push bus in each core. Each branch of the push bus may monitor one core with all the architectural events. All the data collected from the events by the push bus is then sent to a power control unit.

1 Claim, 4 Drawing Sheets



U.S. PATENT DOCUMENTS

7,340,378	B1 *	3/2008	Floyd et al.	702/186
7,454,316	B2 *	11/2008	Bose et al.	702/186
7,516,334	B2 *	4/2009	Suzuoki et al.	713/300
7,598,702	B2 *	10/2009	Walrath	320/107
7,627,784	B1 *	12/2009	Allen et al.	714/30
7,770,034	B2 *	8/2010	Nanja	713/300
7,788,670	B2 *	8/2010	Bodas et al.	718/105
2003/0158609	A1 *	8/2003	Chiu	700/22
2003/0217297	A1 *	11/2003	Gschwind et al.	713/300
2004/0268164	A1 *	12/2004	Flachs et al.	713/320
2004/0268166	A1 *	12/2004	Farkas et al.	713/320
2005/0076251	A1 *	4/2005	Barr et al.	713/300
2005/0120254	A1 *	6/2005	Suzuoki et al.	713/320
2005/0125701	A1 *	6/2005	Hensbergen et al.	713/320
2005/0132238	A1 *	6/2005	Nanja	713/300
2005/4132238		6/2005	Nanja	
2005/0183065	A1 *	8/2005	Wolczko et al.	717/124
2005/0257078	A1 *	11/2005	Bose et al.	714/1
2005/0283677	A1 *	12/2005	Adkisson et al.	714/38
2005/0289362	A1 *	12/2005	Merkin et al.	713/300

2006/0080062	A1 *	4/2006	Bose et al.	702/186
2007/0079294	A1 *	4/2007	Knight et al.	717/130
2007/0168807	A1 *	7/2007	Adkisson	714/731
2007/0180280	A1 *	8/2007	Bolan et al.	713/300
2007/0192641	A1 *	8/2007	Nagendra et al.	713/320
2007/0198863	A1 *	8/2007	Bose et al.	713/300
2007/0294054	A1 *	12/2007	Mericas	702/186
2008/0072110	A1 *	3/2008	Adkisson et al.	714/724
2008/0082844	A1 *	4/2008	Ghiasi et al.	713/323
2008/0301474	A1 *	12/2008	Bussa et al.	713/300

FOREIGN PATENT DOCUMENTS

JP	2005527875	9/2005
WO	2006037119	4/2006
WO	W02006037119	4/2006

OTHER PUBLICATIONS

Japanese Patent Office, Japan, Office Action mailed Sep. 4, 2012 in Japanese application No. 2011-138138.

* cited by examiner

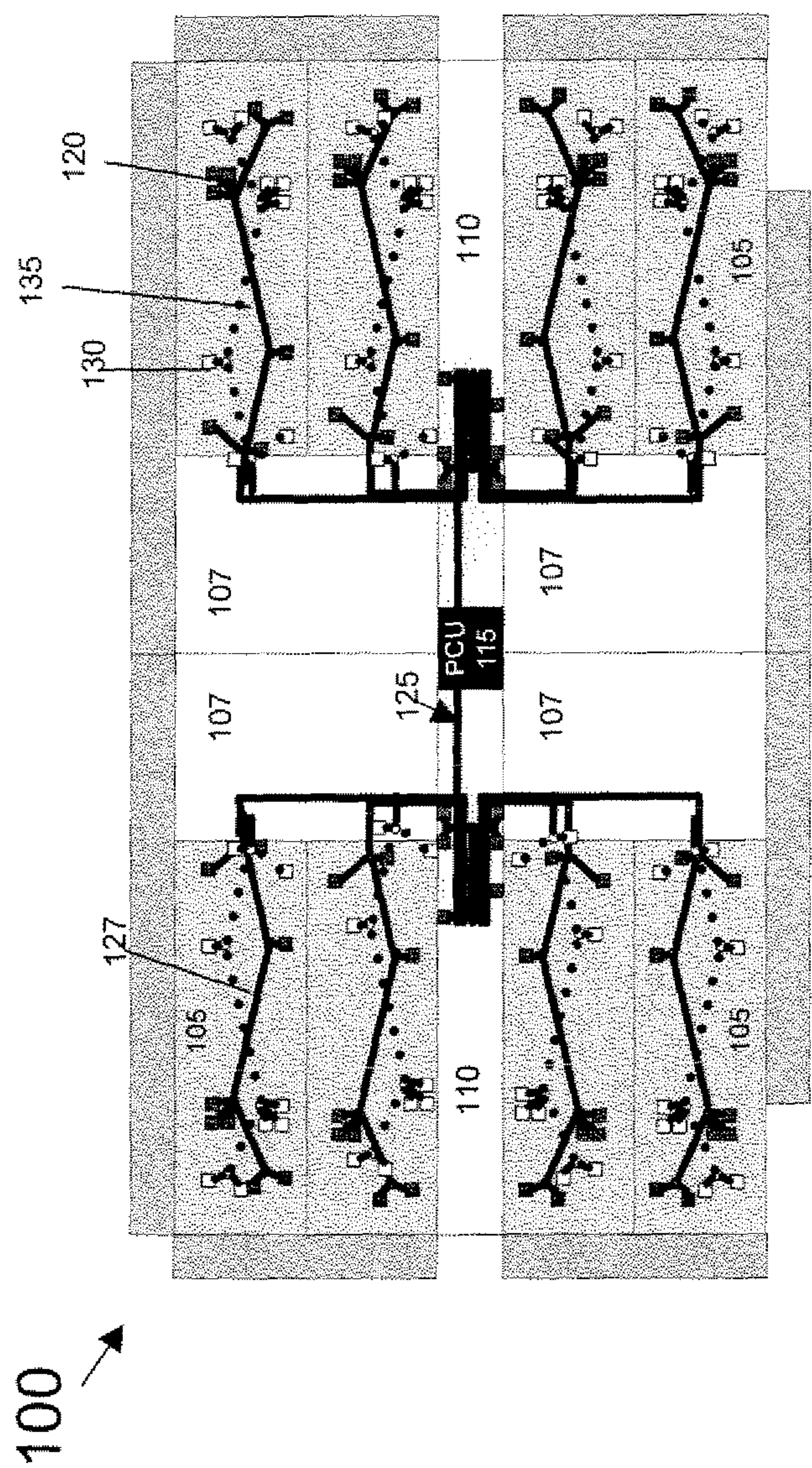


Fig. 1

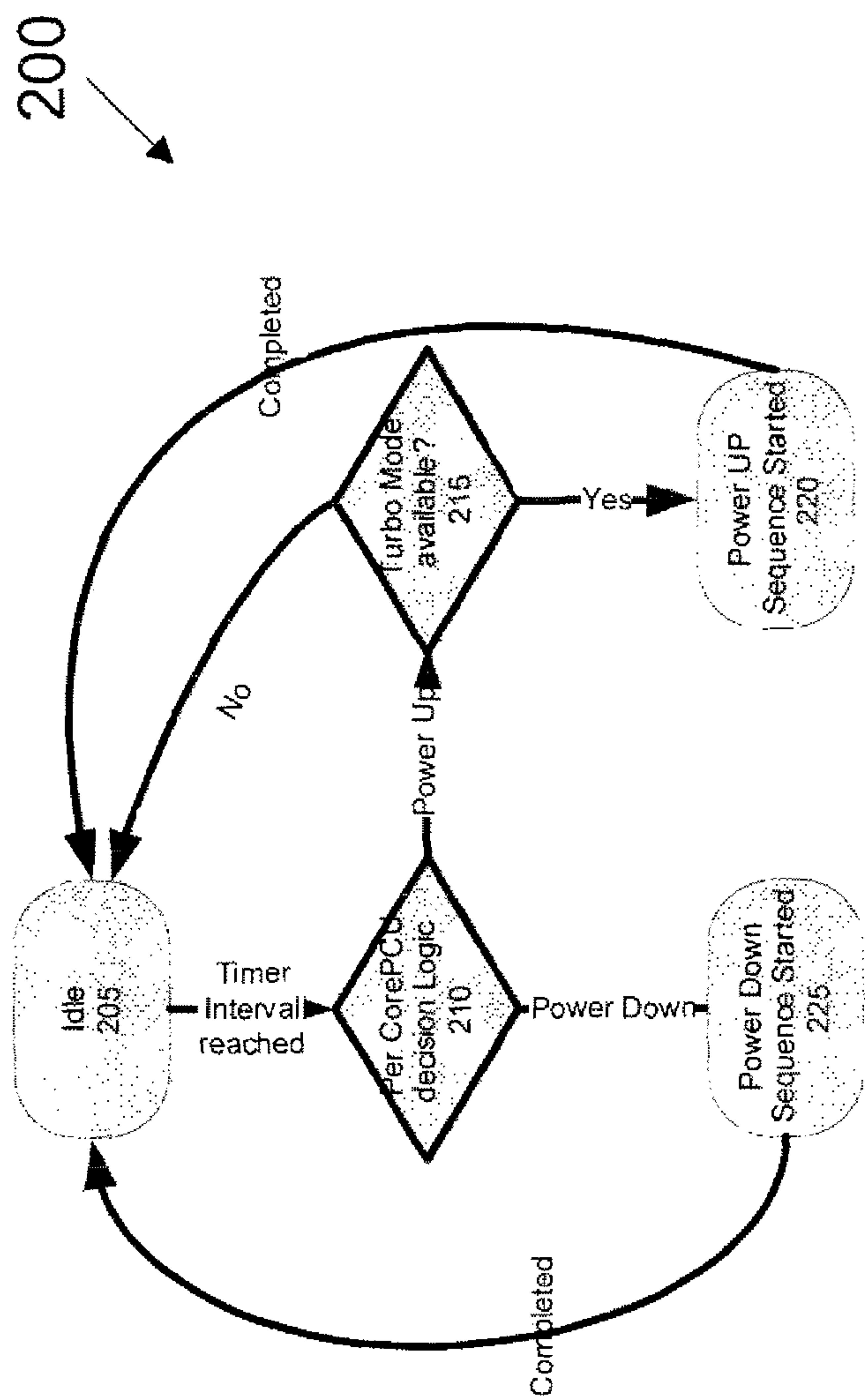


Fig. 2

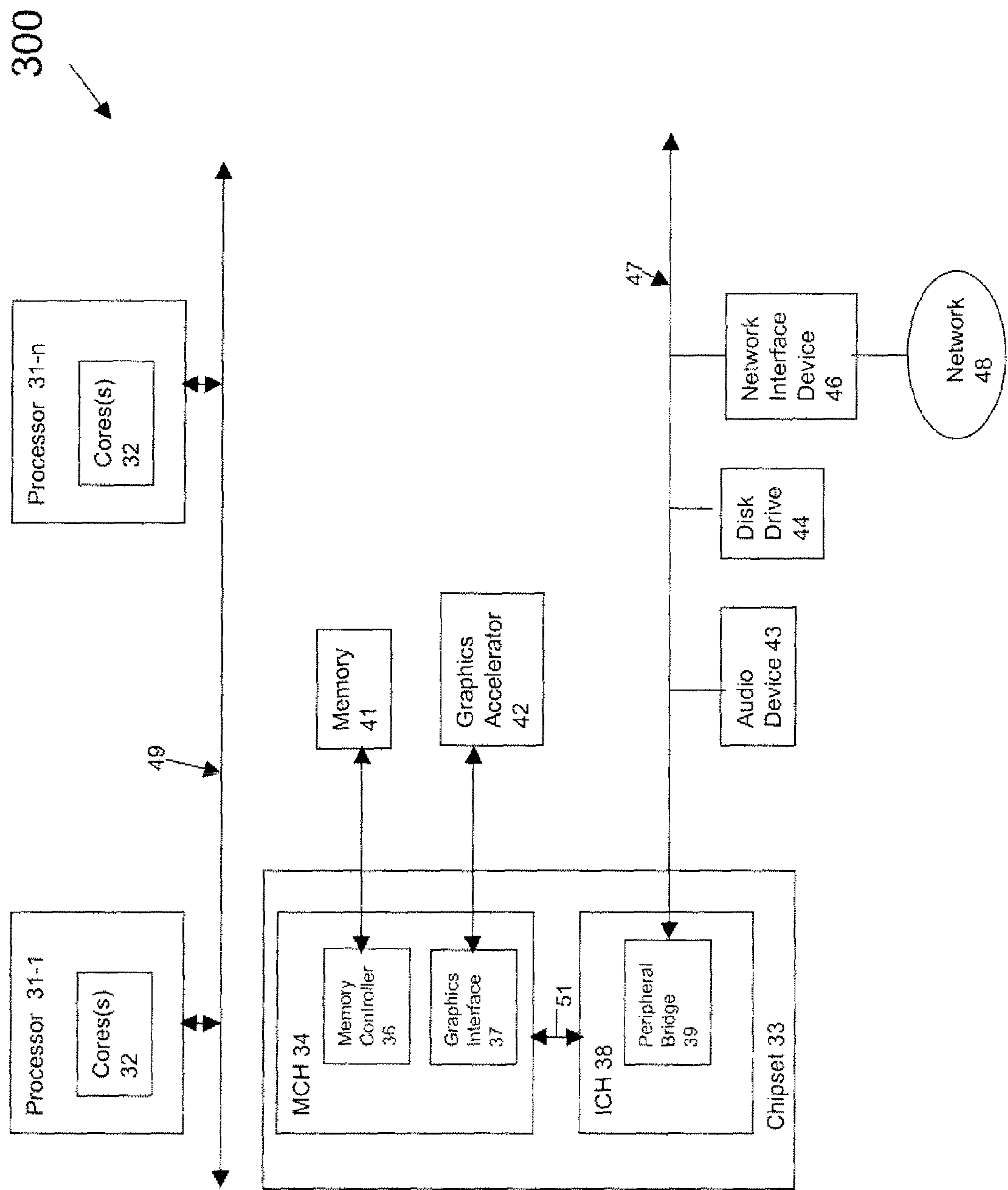
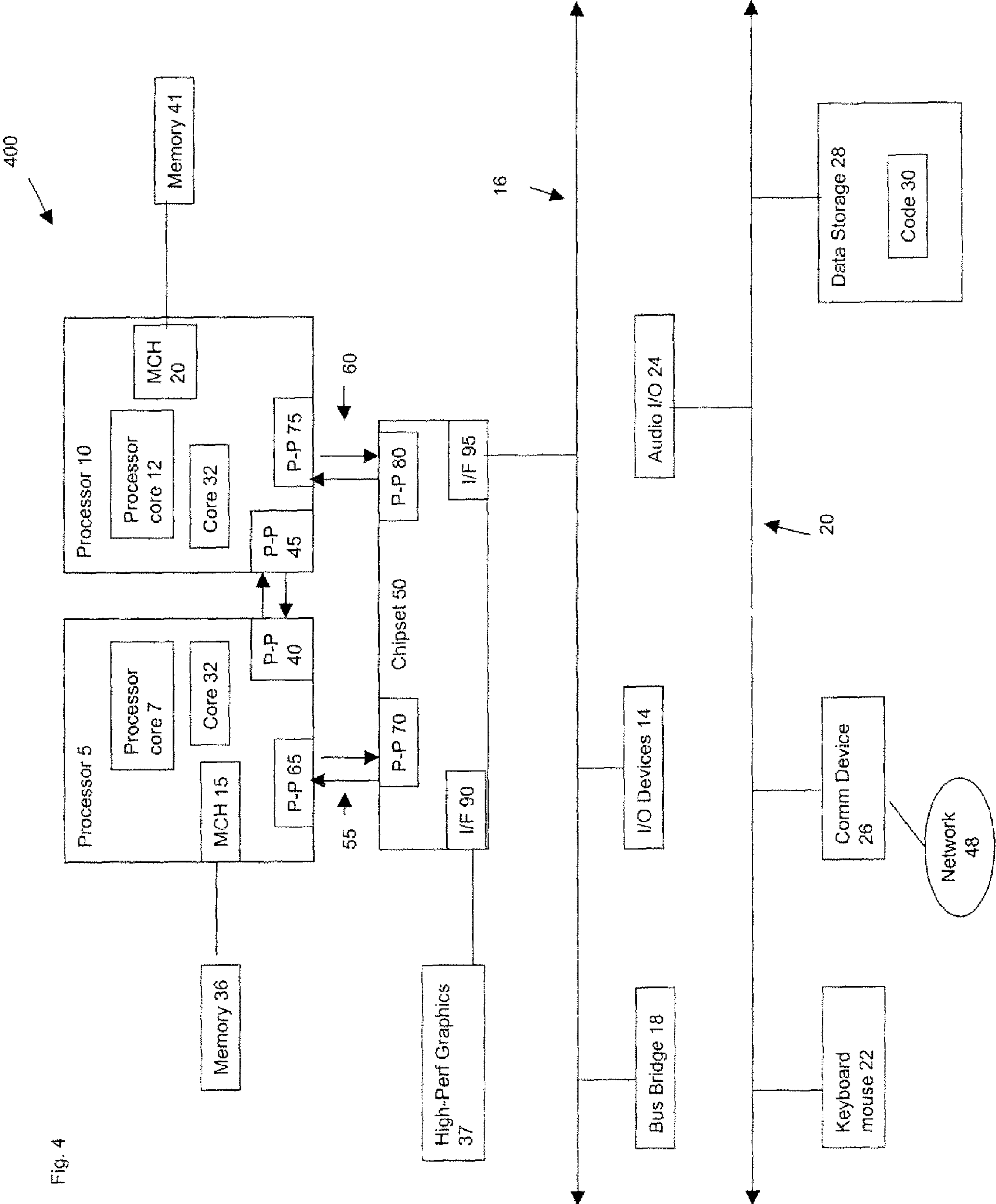


Fig. 3



OPTIMIZING POWER USAGE BY PROCESSOR CORES BASED ON ARCHITECTURAL EVENTS

CLAIM TO PRIORITY

This patent application is a continuation of U.S. patent application Ser. No. 13/329,700 filed on Dec. 19, 2011, entitled "Optimizing Power Usage By Factoring Processor Architectural Events To PMU" which is a continuation of U.S. patent application Ser. No. 11/618,477 filed Dec. 29, 2006, now U.S. Pat. No. 8,117,478 entitled, "Optimizing Power Usage By Processor Cores Based On Architectural Events", all of which are hereby incorporated by reference.

BACKGROUND INFORMATION

Processor power consumption has become a major issue in recent years. The current trend of processor design to multi-core architecture has eased the pressure on power consumption. However, power consumption still grows linearly with the number of cores being designed.

Recent developments in technology have provided new opportunities to further optimize power consumptions. However, these technologies still have several downsides. Specifically, within the new technologies, an Operating System (OS) may fail to detect how effectively the workload is running in the hardware and the associated power cost in the hardware. Secondly, OS interaction may be slow. Thirdly, the current OSes do not control performance power state per core. Finally, turbo mode only takes thermal and sensor feedback into account. It fails to take into account the cores higher performance requirement.

BRIEF DESCRIPTION OF THE DRAWINGS

Various features of the invention will be apparent from the following description of preferred embodiments as illustrated in the accompanying drawings, in which like reference numerals generally refer to the same parts throughout the drawings. The drawings are not necessarily to scale, the emphasis instead being placed upon illustrating the principles of the inventions.

FIG. 1 illustrates a diagram of a thermal sensor and architecture events connection to a PCU in accordance with one embodiment of the invention.

FIG. 2 illustrates a flow chart of a method for architecture event power controller in accordance with one embodiment of the invention.

FIG. 3 is a block diagram of computer system in accordance with an embodiment of the invention.

FIG. 4 is a block diagram of a computing system arranged in a point-to-point configuration, according to one embodiment of the invention.

DETAILED DESCRIPTION

In the following description, for purposes of explanation and not limitation, specific details are set forth such as particular structures, architectures, interfaces, techniques, etc. in order to provide a thorough understanding of the various aspects of the invention. However, it will be apparent to those skilled in the art having the benefit of the present disclosure that the various aspects of the invention may be practiced in other examples that depart from these specific details. In certain instances, descriptions of well-known devices, cir-

cuits, and methods are omitted so as not to obscure the description of the present invention with unnecessary detail.

Some of the embodiments discussed herein may be utilized to perform event handling operations. In an embodiment, an "event" refers to a condition that may or may not require some action to be taken by logic. Furthermore, events may be classified into different types based on the action that is to be taken.

In one embodiment, an "architectural event" refers to an event or condition that may be monitored (e.g., by programming information corresponding to the architectural event into a state). In an embodiment, software may configure a channel to monitor certain architectural event which may not otherwise be observable by software and/or hardware. In an embodiment, an architectural event may generally refer to an event or condition that occurs within processing resources or other logic present on the same integrated circuit chip as a processor.

In an embodiment, architecture events are one factor for making power management decisions. Architecture events are widely used for software performance optimization. By factoring in architecture events for making power management decisions, the embodiment may effectively detect core execution condition that neither OS nor turbo mode can currently detect.

One embodiment provides additional value to a power control unit (PCU) to make appropriate power decisions using existing designs. The PCU handles thermal events, schedules/manages per core's frequency increase/decrease. In the embodiment, logic is added in the PCU. The PCU then takes feedback from counters to assist in making decisions to scale up a particular core.

Architecture events inside a processor are usually detected by a series of physical counters implemented in different areas of the processor. These counters may be referred to as EMON counters. EMON counters are performance monitoring counters that reside in each function unit that counts the number of specific events that has occurred during a period of time.

There may be public EMON counters and private EMON counters. Public EMON counters are those accessible by vendors and users, while private EMON counters are used internally for debugging. Both public and private EMON counters may be used for software and hardware tuning. EMON counters are very useful tools to understand the application behavior and how effective the current application is doing inside the processor.

Instruction retired is one type of EMON counter. By knowing instruction retired per cycle, the system knows how fast the current application is executing inside the processor. More instruction retired per cycle, the better performance the application is getting.

Another type EMON counter is a MLC/LLC cache miss. Programs experience lots of MLC/LLC cache miss usually means they require higher BW to system interconnect. It also means the program will not be effectively executing instruction since most of time the problem will be stalled waiting for data return.

Branch Misprediction is also a type of EMON counter. This is a good indicator to show that the core is not doing lots of "useful" work since it is working on the wrong branch of instruction stream.

TLB miss is also a type of EMON counter. Similar to MLC/LLC miss, a TLB miss usually means stalling the core until data has returned.

There are many more counters defined in any processor. For example, Intel's P4 processor contains more than 200

counters. Some of the counters are for performance tuning while others are for debug/validation purposes. The four counters mentioned above, instruction retired, MLC/LLC cache miss, branch misprediction and TLB miss are crucial to the power management discussed here within.

However, it should be noted that the present disclosure is not limited to the above identified counters. Rather, the present disclosure may take as many architectural events as needed. The present system utilizes a push bus mechanism that can connect all counters serially on the same bus since these counters do not require high bandwidth or high refresh rate.

FIG. 1 illustrates a diagram of a thermal sensor and architecture events connection **100** to a PCU in accordance with one embodiment of the invention. FIG. 1 illustrates a processor with eight cores **105** and four shared last level caches (LLC) **107**. It should be noted that the eight cores and four shared LLCs are shown for illustrative purposes only. Any number of cores or LLCs may be used for this embodiment. A functional unit **110** is shared among the cores **105**. The PCU **115** is one of the units to be shared by the cores **105**.

The dark squares in the cores **105** represent thermal sensors **120**. The cores **105** have thermal sensors **105** to prevent over heating of the cores **105**. A serial push bus **125** connects all the sensors **120** on the bus. The push bus **125** may have various branches. A first branch **127** of the push bus **125** may take up to 96 time slots, where each time slot is reserved for one of the thermal sensors **120**. There is at least one branch **127** of the push bus **125** in each core **105**. Each branch **127** of the push bus **105** may monitor one core **105** with all the thermal sensors **120**. All the data collected from the thermal sensors **120** by the push bus **125** is then sent to the PCU **115**.

A similar approach is made to monitor architectural events by linking different counters via the same serial push bus mechanism and each counter having a designated time slot. The light squares are EMON counters **130**. The dotted line **135** is another branch of the serial push bus. The second branch **135** of the push bus **125** may take up to 96 time slots, where each time slot is reserved for one of the EMON counters. There is at least one branch **135** of the push bus **125** in each core **105**. Each branch **135** of the push bus **125** may monitor one core **105** with all the EMON counters **130**. All the data collected from the EMON counters **130** by the push bus **125** is then sent to the PCU **115**.

The PCU **115** may have very low sampling rate to retrieve information from the push bus **125** which is one reason why all the sensors may share a narrow bus. The serial push bus **125** may not be more than 10 bits wide and runs a low sampling rate. Due to this, there is no high bandwidth requirement to monitor these events. Sampling the counters every 1000 cycles may be sufficient. The sampling frequency should be tuned with the thermal sensors so that PCU **115** decisions may be made coherent between architectural events **130** and thermal sensors **120**.

FIG. 2 illustrates a flow chart of a method **200** for architecture event power controller in accordance with one

embodiment of the invention. The method **200** begins when the system is in idle mode **205**. When each timer interval is reached, architectural event power optimization logic (AEPOL) examines the counts and decides whether a power up or power down sequence should be initiated **210**.

For a power up sequence, AEPOL queries PCU **115** to determine if turbo mode is available **215**. If turbo mode is not available, meaning no reserve power is available, AEPOL goes back to idle mode **205** until the next timer interval arrives. If turbo mode is available, AEPOL initiates a power up sequence **220**. Once the power up sequence is completed AEPOL goes into idle mode **205** until the next timer interval.

For a power down sequence, AEPOL notifies PCU **115** to initiate power down sequence **225**. Once power down sequence is completed, AEPOL goes to idle mode **205** until the next timer interval.

The AEPOL referred to in FIG. 2 is implemented per core. The logic may constantly monitor architectural events to ensure cores are running in it optimal power/performance state.

In one embodiment, architecture event counts of interest are counted and sent to the PCU **115**. PCU **115** then analyzes the counts it receives from each counter and determines whether it should power up or power down a particular core **105**. In one embodiment, if a system interconnect (shared cache, bus control logic, on-die memory control, etc.) also supports power scaling, the same mechanism may be used to scale system interconnect logic.

In an embodiment, all EMON counters **130** monitored should be referenced together to better understand AEPOL behavior. For example, low instruction retired per cycle (IPC) count does not mean core is not running effectively unless we know it also suffers high cache miss or high branch misprediction.

As shown in Table 1 below, the architecture events of interests are grouped together to make decisions. A threshold count for each architecture event counter is also defined. The threshold value is defined based on performance studies for a particular architecture, as performance may differ depending on the architecture of the system. In Table 1, if all related architecture counters are low, this suggests that the system does not suffer from any system bound actions. Therefore, scaling up the frequency on the core does help increasing the performance of the application. On the other hand, if the core is suffering rather high counts of MLC/LLC misses, there will be no benefit to increase core frequency since the performance is bounded by access latency to an off-chip component. In this situation, the system should scale down the core frequency and divert the saved power to other cores that may be doing useful work.

Therefore, AEPOL allows for an intelligent decision making on when to turn on turbo where performance will benefit and can be observed and it lowers the power dissipation with fine tuning performance power state on cores that are stalling on instruction.

TABLE 1

PCU decision tree per core based on architectural event counters						
Architectural Events	Count Threshold	Count Threshold	Count Threshold	Count Threshold	Count Threshold	Count Threshold
Inst. Retired	Low	high	low	low	low	low
MLC Miss	low	don't care	high	high	low	low
LLC Miss	low	don't care	high	low	don't care	low

TABLE 1-continued

PCU decision tree per core based on architectural event counters						
Architectural Events	Count Threshold	Count Threshold	Count Threshold	Count Threshold	Count Threshold	Count Threshold
TLB Miss	low	don't care	don't care	don't care	don't care	high
Branch Miss	low	don't care	don't care	high	high	don't care
Final Decision	PowerUP	Power UP	PowerDown	PowerDown	Power Up	PowerDown

There can be many more scenarios or other useful architectural event counters that can do finer grain control of the core power usage and the AEPOL logic is applicable to all these situations.

With AEPOL optimization, the system is given a smarter way to utilize the available power to the processor. In future processor design where power aware architecture must exist across the board, AEPOL not only provides a way to conserve energy when it is not needed, it also provides opportunity to optimize performance further within the same power budget. This mechanism opens a whole new range of opportunity for finer grain hardware tuning.

FIG. 3 illustrates a block diagram of a computing system 300 in accordance with an embodiment of the invention. The computing system 300 may include one or more central processing units(s) (CPUs) 31 or processors that communicate via an interconnection network (or bus) 49. The processors 31 may be any type of a processor such as a general purpose processor, a network processor (that processes data communicated over a computer network 48, or other types of a processor (including a reduced instruction set computer (RISC) processor or a complex instruction set computer (CISC). Moreover, the processors 31 may have a single or multiple core design. The processors 31 with a multiple core design may integrate different types of processor cores on the same integrated circuit (IC) die. Also, the processors 31 may utilize the embodiments discussed with reference to FIGS. 1 and 2. For example, one or more of the processors 31 may include one or more processor cores 32. Also, the operations discussed with reference to FIGS. 1 and 2 may be performed by one or more components of the system 300.

A chipset 33 may also communicate with the interconnection network 49. The chipset 33 may include a memory control hub (MCH) 34. The MCH 34 may include a memory controller 36 that communicates with a memory 41. The memory 41 may store data and sequences of instructions that are executed by the CPU 31, or any other device included in the computing system 300. In one embodiment of the invention, the memory 41 may include one or more volatile storage (or memory) devices such as random access memory (RAM), dynamic RAM, (DRAM), synchronous DRAM (SDRAM), static RAM (SRAM), or the like. Nonvolatile memory may also be utilized such as a hard disk. Additional devices may communicate via the interconnection network 49, such as multiple CPUs and/or multiple system memories.

The MCH 34 may also include a graphics interface 37 that communicates with a graphics accelerator 42. In one embodiment of the invention, the graphics interface 37 may communicate with the graphics accelerator 42 via an accelerated graphics port (AGP). In an embodiment of the invention, a display (such as a flat panel display) may communicate with the graphics interface 37 through, for example, a signal converter that translates a digital representation of an image stored in a storage device such as a video memory or system memory into display signals that are interpreted and displayed

by the display. The display signals produced by the display device may pass through various control devices before being interpreted by and subsequently displayed on the display.

A hub interface 51 may allow the MCH 34 to communicate with an input/output control hub (ICH) 38. The ICH 38 may provide an interface to I/O devices that communicate with components of the computing system 300. The ICH 38 may communicate with a bus 47 through a peripheral bridge (or controller) 39, such as a peripheral component interconnect (PCI) bridge, a universal serial bus (USB) controller, or the like. The bridge 39 may provide a data path between the CPU 31 and peripheral devices. Other types of topologies may be utilized. Also, multiple buses may communicate with the ICH 38, e.g. through multiple bridges or controllers. Moreover, other peripheral in communication with the ICH 38 may include, in various embodiments of the invention, integrated drive electronics (IDE) or small computer system interfaces (SCSI) hard drive(s), USB ports, a keyboard, a mouse, parallel port(s), serial port(s), floppy disk drive(s), digital output support (e.g., digital video interface (DVI)), or the like.

The bus 47 may communicate with an audio device 43, one or more disk drive(s) 44, and a network interface device 46 (which communicates with the computer network 48). Other devices may be in communication with the bus 47. Also, various components (such as the network interface device 46) may be in communication with the MCH 34 in some embodiments of the invention. In addition, the processor 31 and the MCH 34 may be combined to form a single chip. Furthermore, the graphics accelerator 42 may be included within the MCH 34 in other embodiments of the invention.

Furthermore, the computing system 300 may include volatile and/or nonvolatile memory (or storage). For example, nonvolatile memory may include one or more of the following: read-only memory (ROM), programmable ROM (PROM), erasable PROM (EPROM), electrically EPROM (EEPROM), a disk drive (e.g., 44) a floppy disk, a compact disk ROM (CD-ROM), a digital versatile disk (DVD), flash memory, a magneto-optical disk, or other types of nonvolatile machine-readable media capable of storing electronic instructions and/or data.

FIG. 4 illustrates a computing system 400 that is arranged in a point-to-point (PtP) configuration, according to an embodiment of the invention. In particular, FIG. 4 shows a system where processors, memory, and input/output devices are interconnected by a number of point to point interfaces. The operations discussed with reference to FIGS. 1-3 may be performed by one or more components of the system 400.

As illustrated in FIG. 4, the system 400 may include several processors, of which only two, processors 5, 10 are shown for clarity. The processors 5, 10 may each include a local memory controller hub (MCH) 15, 20 to allow communication with memories 15, 20. The memories 15, and/or 20 may store various data such as those discussed with reference to the memory 512.

7

The processors **5**, **10** may be any type of a processor such as those discussed with reference to the processors **31** of FIG. **3**. The processors **5**, **10** may exchange data via a point-to-point interface **93** using PtP interface circuits **40** and **45**, respectively. The processors **5**, **10** may each exchange data with a chipset **50** via individual PtP interfaces **55**, **60** using point to point interface circuits **65**, **70**, **75**, **80**. The chipset **50** may also exchange data with a high-performance graphics circuit **37** via a high performance graphic interface **97**, using a PtP interface circuit **90**.

At least one embodiment of the invention may be provided within the processors **5**, **10**. For example, one or more of the processor core(s) **32** may be located within the processors **5**, **10**. Other embodiments of the invention, however, may exist in other circuits, logic units, or devices with the system **400** of FIG. **6**. Furthermore, other embodiments of the invention may be distributed throughout several circuits, logic units, or devices illustrated in FIG. **4**.

The chipset **50** may communicate with a bus **16** using a PtP interface circuit **95**. The bus **16** may have one or more devices that communicate with it, such as a bus bridge **18** and **110** devices **14**. Via a bus **20**, the bus bridge **14** may be in communication with other devices such as a keyboard/mouse **22**, communication devices **26** (such as modems, network interface devices, etc. that may be in communication with the computer network **48**), audio I/O devices, and/or a data storage device **28**. The data storage device **28** may store code **30** that may be executed by the processors **5** and/or **10**.

In various embodiments of the invention, the operations discussed herein, e.g., with reference to FIGS. **1-4** may be implemented by hardware (e.g., circuitry), software, firmware, microcode, or combinations thereof, which may be provided as a computer program product, e.g., including a machine readable or computer readable medium having stored thereon instructions (or software procedures) used to program a computer to perform a process discussed herein. Also, the term “logic” may include, by way of example, software, hardware, or combinations of software and hardware. The machine readable medium may include a storage device such as those discussed with respect to FIGS. **1-4**.

Additionally, such computer readable media may be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., client) by way of data signals embodied in a propagation medium via a communication link

8

(e.g., a bus, a modem, or a network connection). Accordingly, herein, a carrier wave shall be regarded as comprising a machine readable medium.

Reference in the specification to “one embodiment” or “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least an implementation. The appearances of the phrase “in one embodiment” in various places in the specification may not be all referring to the same embodiment.

Also, in the description and claims, the term “coupled” and “connected”, along with their derivatives, may be used. In some embodiments of the invention, “connected” may be used to indicate that two or more elements are in direct physical or electrical contact with each other. “Coupled” may mean that two or more elements are in direct physical or electrical contact. However, “coupled” may also mean that two or more elements may not be in direct contact with each other, but may still cooperate or interact with each other.

Thus, although embodiments have been described in language specific to structural features and/or methodological acts, it is to be understood that claimed subject matter may not be limited to the specific features or acts described. Rather, the specific features and acts are disclosed as sample forms of implementing the claimed subject matter.

What is claimed is:

1. A processor comprising:

eight processing cores;

one or more caches;

a power control unit;

a plurality of thermal sensors, wherein the plurality of thermal sensors are distributed within the processor to collect thermal data of the processor,

a plurality of event monitoring counters to monitor one or more events in the processor, wherein the one or more events include instructions retired per cycle, cache miss, stalls, and branch mis-prediction, and

a bus to couple the plurality of event monitoring counters and the plurality of thermal sensors to the power control unit, wherein the bus is to transfer data including the thermal data and the one or more events to the power control unit,

wherein the power control unit is to manage the power consumed by the processor based on the data received on the bus.

* * * * *